An Efficient Algorithm for Approximate Betweenness Centrality Computation

Mostafa Haghir Chehreghani
Department of Computer Science, KU Leuven
Celestijnenlaan 200a - box 2402
3001 Leuven, Belgium
mostafa.haghirchehreghani@cs.kuleuven.be

ABSTRACT

Betweenness centrality is an important centrality measure widely used in social network analysis, route planning etc. However, even for mid-size networks, it is practically intractable to compute exact betweenness scores. In this paper, we propose a generic randomized framework for unbiased approximation of betweenness centrality. The proposed framework can be adapted with different sampling techniques and give diverse methods. We discuss the conditions a promising sampling technique should satisfy to minimize the approximation error and present a sampling method partially satisfying the conditions. We perform extensive experiments and show the high efficiency and accuracy of the proposed method.

Categories and Subject Descriptors

G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms, Path and circuit problems*; E.1 [**Data**]: Data structures—*Graphs and networks*

General Terms

Theory

Keywords

Centrality, betweenness centrality, social network analysis, approximate algorithms.

1. INTRODUCTION

Betweenness centrality of a vertex, introduced by Linton Freeman [6], is defined as the number of shortest paths (geodesic paths) from all (source) vertices to all others that pass through that vertex. He used it as a measure for quantifying the control of a human on the communication between other humans in a social network [6]. Betweenness centrality is also used in some well-known algorithms for clustering and community detection in social and information networks [8].

Although betweenness centrality computation is tractable in theory in the sense that there exist polynomial time and space algorithms, the most efficient existing exact method is Brandes's algorithm [3] which requires $O(nm)$ time for unweighted graphs and $O(nm + n^2 \log n)$ time for weighted graphs (n is the number of vertices and m is the number of edges in the network). Therefore, exact betweenness centrality computation is not practically applicable, even for mid-size networks. The next bad news is that computing exact betweenness centrality of a single vertex is not easier than computing betweenness centrality of all vertices. Therefore, the above mentioned worst case bounds hold if someone wants to compute betweenness centrality of one or a few vertices. However, in many applications it might be required to compute betweenness centrality of only one or a few vertices. For example, the index might be computed only for core vertices of communities in social/information networks [12], or hubs in communication networks.

To make betweenness centrality practically computable, in recent years, several algorithms have been proposed for approximate betweenness centrality computation [4], [1] and [7]. Existing algorithms fall into one of the following categories.

1. Some algorithms like [4] and [7] try to approximate betweenness centrality of all vertices in the network. For such methods the value computed for every vertex is not of high importance. Instead, the main goal is to correctly estimate the relative rank of all vertices.

2. Some others, like the method presented in [1], aim to approximate betweenness centrality of a single vertex (or a few vertices) in time faster than computing betweenness centrality of all vertices. For such methods, the accuracy of the estimated betweenness centrality is important.

Our focus in this paper is the second category of algorithms, i.e. we aim at developing an efficient and accurate algorithm for betweenness centrality computation of a single vertex. In this paper, we propose a generic randomized framework for unbiased approximation of betweenness centrality. In the proposed framework, a source vertex i is selected by some strategy, single-source betweenness scores of all vertices on i are computed, and the scores are scaled as estimations of betweenness centralities. While our method might seem similar to the method of Brandes and Pich [4], they have a key difference. In the method of [4], for a few source vertices, single-source betweenness scores are computed and for the rest, they are *extrapolated*. Betweenness centralities are the sum of all single-source betweenness scores (which are either computed or extrapolated). In our method, single-source betweenness scores are computed for one single source chosen randomly, and the obtained scores are *scaled* as estimations of betweenness centralities.

We discuss the condition a promising sampling technique should satisfy to minimize the approximation error for a single vertex.

CIKM'13 October 27-November 1, 2013, Burlingame, CA, USA.

http://dx.doi.org/10.1145/2505515.2507826.

Since it might be computationally expensive to find such a sampling, we propose a sampling technique which partially satisfies the condition. While the algorithm of [1] is intuitively presented for high centrality vertices, in our method, the sampling technique can be revised to optimize itself for both high centrality vertices and low centrality vertices. The proposed method can be used to compute similar centrality notions like *stress centrality*, which is also based on counting shortest paths. We perform extensive experiments on real-world networks from different domains, and show that compared to existing exact and inexact algorithms, our method works with higher accuracy or it gives significant speedups.

The rest of this paper is organized as follows. In Section 2, preliminaries and definitions related to betweenness centrality computation are given. In Section 3, we present a generic randomized algorithm for betweenness centrality computation. In Section 4, we discuss the sampling methods. We empirically evaluate the proposed method in Section 5 and show its efficiency and high accuracy. Finally, the paper is concluded in Section 6.

2. PRELIMINARIES

Throughout the paper, $\mathbf{G}$ refers to a graph (network). For simplicity, we suppose $\mathbf{G}$ is a connected and loop-free graph without multi-edges. Throughout the paper, we assume $\mathbf{G}$ is an unweighted graph, unless it is explicitly mentioned that $\mathbf{G}$ is weighted. $V(\mathbf{G})$ and $E(\mathbf{G})$ refer to the set of vertices and the set of edges of $\mathbf{G}$, respectively. Throughout the paper, n points to $|V(\mathbf{G})|$ and m points to $|E(\mathbf{G})|$. For an edge $e = (u, v) \in E(\mathbf{G})$, u and v are two endpoints of e. A *shortest path* (also called a *geodesic path*) between two vertices $u, v \in V(\mathbf{G})$ is a path whose size is minimum, among all paths between u and v. For two vertices $u, v \in V(\mathbf{G})$, we use $d(u, v)$, to denote the size (the number of edges) of a shortest path connecting u and v. By definition, $d(u, v) = 0$ and $d(u, v) = d(v, u)$. For $s, t \in V(\mathbf{G})$, σ_{st} denotes the number of shortest paths between s and t, and $\sigma_{st}(v)$ denotes the number of shortest paths between s and t that also pass through v. We have $\sigma_s(v) = \sum_{t \in V(\mathbf{G}) \setminus \{s,v\}} \sigma_{st}(v)$. *Betweenness centrality* of a vertex v is defined as:

$$BC(v) = \sum_{s,t \in V(G) \setminus \{v\}} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{1}$$

A notion which is widely used for counting the number of shortest paths in a graph is the directed acyclic graph (DAG) containing all shortest paths starting from a vertex s (see e.g. [3]). In this paper, we refer to it as the *shortest-path-DAG*, or *SPD* in short, rooted at s. For every vertex s in a graph G, the *SPD* rooted at s is unique, and it can be computed in $O(m)$ time for unweighted graphs and in $O(m + n \log n)$ time for weighted graphs [3]. In [3], the authors introduced the notion of the *dependency score* of a vertex $s \in V(\mathbf{G})$ on a vertex $x \in V(\mathbf{G}) \setminus \{s\}$, which is defined as:

$$\delta_{s\bullet}(v) = \sum_{t \in V(\mathbf{G}) \setminus \{v,s\}} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{2}$$

We have: $BC(v) = \sum_{s \in V(\mathbf{G}) \setminus \{v\}} \delta_{s\bullet}(v)$. As mentioned in [3], given the SPD rooted at s, dependency scores of s on all other vertices can be computed in $O(m)$ time.

3. APPROXIMATE BETWEENNESS CENTRALITY COMPUTATION

Algorithm 1 shows the high level pseudo code of the algorithm proposed for approximate betweenness centrality computation. First the following probabilities are computed

$$p_1, p_2, \ldots, p_n \geq 0 \text{ such that } \sum_{i=1}^{n} p_i = 1 \tag{3}$$

Then, at every iteration t of the loop in Lines 8-15 of Algorithm 1: (I) an $i \in \{1, \ldots, n\}$ is selected with probability p_i, (II) the SPD rooted at i is computed, (III) dependency score of vertex v on i, $\delta_{i\bullet}(v)$, is computed, and (IV) $\frac{\delta_{i\bullet}(v)}{p_i}$ is the estimation of $BC(v)$ in iteration t. The average of betweenness centralities calculated in different iterations is returned as the estimation of betweenness centrality.

Algorithm 1 High level pseudo code of the algorithm of approximate betweenness centrality computation.

```
1: ApproximateBetweenness
2: Require. A network (graph) G, the number of samples T.
3: Ensure. Betweenness centrality of vertices of G.
4: Compute probabilities p_1, ..., p_n
5: for all vertices v ∈ V(G) do
6:     B[v] ← 0
7: end for
8: for all t = 1 to T do
9:     Select a vertex i with probability p_i
10:    Form the SPD D rooted at i
11:    Compute dependency scores of every vertex v on i
12:    for all vertex v ∈ V(G) do
13:        B[v] ← B[v] + δ_{i•}(v) / p_i
14:    end for
15: end for
16: for all i ∈ {1, ..., n} do
17:    B[i] ← B[i] / T
18: end for
19: return B
```

LEMMA 1. *In Algorithm 1, for every vertex v we have*

$$\mathbf{E}(B[v]) = BC(v) \tag{4}$$

and

$$\mathbf{Var}(B[v]) = \frac{1}{T} \sum_{i=1}^{n} \left(\frac{\delta_{i\bullet}(v)^2}{p_i} \right) - \frac{BC(v)^2}{T} \tag{5}$$

PROOF. We have:

$$\mathbf{E}(B[v]) = \frac{T \sum_{i=1}^{n} \frac{p_i \delta_{i\bullet}(v)}{p_i}}{T} = BC(v)$$

and

$$\mathbf{Var}(B_t[v]) = \mathbf{E}(B_t[v]^2) - \mathbf{E}(B_t[v])^2 = \sum_{i=1}^{n} \frac{\delta_{i\bullet}(v)^2}{p_i} - BC(v)^2$$

Since $B[v]$ is the average of T independent copies of $B_t[v]$, therefore

$$\mathbf{Var}(B[v]) = \frac{1}{T} \sum_{i=1}^{n} \left(\frac{\delta_{i\bullet}(v)^2}{p_i} \right) - \frac{BC(v)^2}{T} \tag{6}$$

□

Some existing algorithms can be described as adaptations of Algorithm 1 with specific sampling methods. For example, if vertices i are selected uniformly at random (i.e. $p_i = \frac{1}{n}$ for $1 \leq i \leq n$), then it will give the randomized algorithm presented in [1]. We

note that instead of taking T samples, we can define a criteria for the termination of the loop in Lines 8-15 of Algorithm 1. For example, similar to the algorithm of [1], we can stop when $B[v] \geq cn$ for some constant c.

4. SAMPLING METHODS

Suppose that we want to approximate betweenness centrality of a vertex v. The following Lemma defines the probabilities minimizing the approximation error.

LEMMA 2. *If in Algorithm 1 source vertices i are selected with probabilities*

$$p_i = \frac{\delta_{\bullet i}(v)}{\sum_{j=1}^{n} \delta_{\bullet j}(v)} \tag{7}$$

the approximation error (i.e. variance of $B[v]$) is minimized. In this case, variance of $B[v]$ will be 0 [1].

PROOF. Omitted due to lack of space. □

Therefore, using probabilities p_i defined in Equation 7, gives an exact method in the sense that it makes the approximation error 0. However, time complexity of computing optimal p_i's is the same as exact betweenness centrality computation. Although it is not practically efficient to use probabilities p_i defined in Equation 7, they can help us to define properties of a good sampling. From Equation 7, we can conclude that in a good sampling, for every two vertices i and i', the following must hold:

$$p_i \geq p_{i'} \Leftrightarrow \delta_{\bullet i}(v) \geq \delta_{\bullet i'}(v) \tag{8}$$

which means vertices with higher dependency scores on v, must be selected with a higher probability.

However, finding probabilities $p_1, p_2, \ldots, p_n$ which satisfy Equation 8 might be computationally expensive, since it needs to compute dependency scores of all vertices on v which is as bad as computing dependency scores of every source vertex on all vertices. In order to design practically efficient sampling techniques, we consider relaxations of Equation 8. Consider two vertices i and i' such that $d(i', v) > d(i, v)$. If in the SPD rooted at v there exists an ancestor-descendant relationship between i and i' and i is the only ancestor of i' at the level $d(i, v)$, then, it can be shown that for $k \in \{i, i'\}$, probability p_k defined as

$$p_k = \frac{\frac{1}{d(k,v)}}{\sum_{j=1}^{n} \frac{1}{d(j,v)}} \tag{9}$$

satisfies Equation 8.

The positive aspect of the sampling technique presented in Equation 9 is that it only needs to compute the distance between vertex v and every vertex in the graph: the single-source shortest path, or SSSP in short, problem. For unweighted graphs, this problem can be solved in $O(m)$ time and for weighted graphs, using Fibonacci heap, it is solvable in $O(m + n \log n)$ time [5]. It means that the sampling method presented in Equation 9 is practically efficient.

[1] What this lemma suggests, somehow contradicts the source vertex selection procedure presented in [7]. In the method of [7] the scheme for aggregating dependency scores changes so that vertices do not profit from being near the selected source vertices. However, Lemma 2 says it is better to select source vertices based on their dependency scores on v, and as we will see later, it might result in preferring source vertices which are closer to v.
The reason of this contradiction is that while here we aim at precisely approximating betweenness centrality of *some specific vertex* v, the method of [7] aims to rank *all vertices* based on their betweenness scores.

Therefore, with probabilities p_i defined in Equation 9, a vertex i is selected and dependency score of i on v is computed, and the result is scaled. For unweighted graphs, it gives an $O(Tm)$ time algorithm for approximate betweenness centrality computation. For weighted graphs (with positive weights), time complexity of the approximation algorithm will be $O(Tm + Tn \log n)$.

5. EXPERIMENTAL RESULTS

Our experiments were done on one core of a single AMD Processor 270 clocked at 2.0 GHz with 8 GB main memory and 2×1 MB L2 cache, running Ubuntu Linux 12.0. The program was compiled by the GNU C++ compiler 4.0.2 using optimization level 3. We compare our proposed method with the algorithm presented in [1]. As mentioned earlier, methods like [4] and [7] aim to rank vertices based on betweenness scores (and the betweenness score of an individual vertex is not very important for them). Therefore, they are not suitable for our comparisons. We refer to the algorithm of [1] as *uniform sampling*, since it chooses source vertices uniformly at random. We refer to our proposed method as *distance-based sampling*. We also compare the methods with Brandes's algorithm for exact betweenness centrality computation [3].

We performed extensive experiments on different real-world networks to assess the quantitative and qualitative behavior of the proposed algorithm. We used two DBLP citation networks dblp0305 and dblp0507 [2], the Wiki-Vote social network [9], the Enron-Email communication network [10], and the CA-CondMat collaboration network [11]. Table 1 summarizes specifications of our real-world networks.

Table 1: Summary of real-world networks.

Dataset	# vertices	# edges	Avg. degree
dblp-0305	109,044	233,961	4.29
dblp-0507	135,116	290,363	4.28
Enron-Email	36,692	367,662	20.04
Wiki-Vote	7,115	103,689	29.14
CA-CondMat	23,133	93,497	8.08

For a vertex v, the empirical betweenness approximation error (which is reported in the experiments) is defined as:

$$err(v) = \frac{|App(v) - BC(v)|}{BC(v)} \times 100 \tag{10}$$

where $App(v)$ is the computed approximate score.

In our experiments, we consider several vertices of a dataset and for every vertex, we compute distance-based probabilities, exact betweenness scores, and approximate betweenness scores using distance-based and uniform samplings. Table 2 summarizes the average results (i.e. the sum of results of all vertices divided by their number) obtained for different datasets. In all experiments, for both uniform and distance-based samplings, the number of samples (the number of source vertices selected) is 10% of the number of vertices in the network. Therefore, the running time of the approximate methods is around 10% of the running time of the exact method.

The first dataset is Wiki-Vote which is dense (its average degree is 29.14). For most of vertices of Wiki-Vote, distance-based sampling gives a better approximation. The extra time needed by the distance-based sampling to compute required shortest path distances is quite tiny and ignorable compared to running time of the whole process. Since for all datasets it is a tiny value varying in different runs, we report an upper bound for it. For the Wiki-Vote

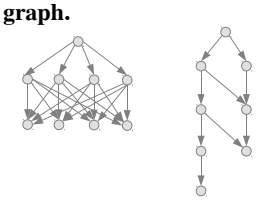
Table 2: Comparing average approximation error and average running time of uniform sampling, distance-based sampling, and exact method, for single vertices in different datasets.

Database	Exact BC		Approximate BC				
			Distance-based sampling		Uniform sampling		
	Avg. BC score	Avg. time (sec)	Avg. error (%)	Avg. dist. comp. time (sec)	Avg. error (%)	Avg. time (sec)	# iterations
Wiki Vote	76056.85	515.09	**37.0**%	< 1	**41.13**%	46.05	10%
Email-Enron	2775100.8	9033.11	**15.75**%	< 1	**25.28**%	925.80	10%
dblp0305	564246.41	19149.8	**7.59**%	< 2	**64.73**%	1747.15	10%
dblp0507	798125.00	35140	**7.19**%	< 2	**50.17**%	2863.82	10%
CA-CondMat	691667	3026.9	**10.8**%	< 1	**20.81**%	315.3	10%

dataset, it is always smaller than 1 second. The next dataset is Email-Enron. Compared to Wiki-Vote, it is less dense (but still dense) and larger. Over this dataset, the approximation error of distance-based sampling is better than the uniform sampling.

Dblp0305 and dblp0507 are large and relatively sparse datasets. As reflected in Table 2, over these datasets, distance-based sampling works much better than uniform sampling. This means that on sparse datasets, the difference between the approximation quality of two methods is more considerable. It has several reasons. The first reason is that in very dense datasets, many vertices have the same (and small) distance from v (v is the vertex whose betweenness centrality is approximated). Therefore, distance-based sampling becomes closer to the uniform sampling. The second reason is that in sparse networks, in the SPD rooted at v, the probability that a vertex i has only one ancestor at some level k is lower than this probability in dense graphs. Figure 1 compares these two situations. It means that in sparse networks, distance-based sampling is closer to the optimal sampling, because by distance-based sampling, a larger number of vertices will satisfy the condition expressed in Equation 7. As a result, on sparse networks, distance-based sampling becomes much more effective than uniform sampling.

Figure 1: In sparse graphs, distance-based sampling is closer to optimal sampling. The graph in the left side shows a SPD in a dense graph, and the graph in the right side shows a SPD in a sparse graph.



Finally, the methods were compared on the CA-CondMat dataset which contains scientific collaborations between authors of papers submitted to Condense Matter category [11]. The average degree in this dataset is 8.08 which means it is denser than dblp0305 and dblp0507, but less dense than Wiki-Vote and Email-Enron. Over this dataset, the approximation error of uniform sampling is almost twice of the approximation error of distance-based sampling.

6. CONCLUSION

In this paper, we presented a generic randomized framework for unbiased approximation of betweenness centrality. In the proposed framework, a source vertex i is selected by some strategy, single-source betweenness scores of all vertices on i are computed, and the scores are scaled as estimations of betweenness centralities. Our proposed framework can be adapted with different sampling techniques to give diverse methods for approximating betweenness centrality. We discussed the conditions a promising sampling technique should satisfy to minimize the approximation error, and proposed a sampling technique which partially satisfies the conditions. Our experiments show the high efficiency and quality of the proposed method.

7. ACKNOWLEDGMENTS

This work was partially supported by ERC Starting Grant 240186 "MiGraNT: Mining Graphs and Networks: a Theory-based approach".

8. REFERENCES

[1] D. A. Bader, S. Kintali, K. Madduri and M. Mihail, Approximating Betweenness Centrality, WAW, 124-137, 2007.

[2] M. Berlingerio, F. Bonchi, B. Bringmann and A. Gionis, Mining Graph Evolution Rules, ECML/PKDD (1), 115-130, 2009.

[3] U. Brandes. A Faster Algorithm for Betweenness Centrality. Journal of Mathematical Sociology, 25(2), 163-177, 2001.

[4] U. Brandes, C. Pich, Centrality estimation in large networks, Intl. Journal of Bifurcation and Chaos, 17(7), 303-2318, 2007.

[5] M. L. Fredman and R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, Journal of the Association for Computing Machinery, 34(3), 596-615, 1987.

[6] L. C. Freeman, A set of measures of centrality based upon betweenness, Sociometry, 40, 35-41, 1977.

[7] R. Geisberger, P. Sanders, D. Schultes, Better Approximation of Betweenness Centrality, ALENEX, 90-100, 2008.

[8] M. Girvan and M. E. J. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. USA 99, 7821-7826, 2002.

[9] J. Leskovec, D. Huttenlocher, J. Kleinberg, Signed Networks in Social Media, CHI 2010.

[10] J. Leskovec, K. Lang, A. Dasgupta, M. Mahoney, Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters, Internet Mathematics, 6(1), 29-123, 2009.

[11] J. Leskovec, J. Kleinberg, C. Faloutsos, Graph Evolution: Densification and Shrinking Diameters. ACM Transactions on Knowledge Discovery from Data (ACM TKDD), 1(1), 2007.

[12] Y. Wang, Z. Di, Y. Fan, Identifying and Characterizing Nodes Important to Community Structure Using the Spectrum of the Graph, PLoS ONE 6(11), e27418, 2011.